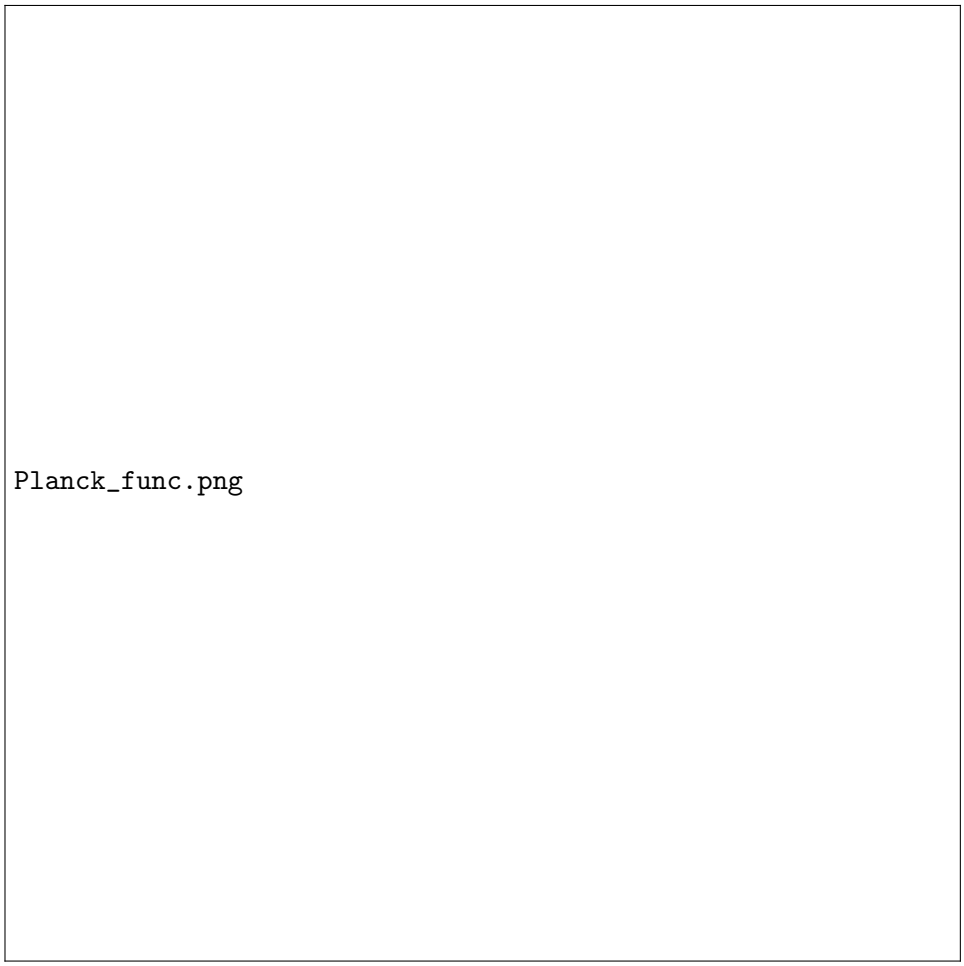
```
"""Some helper functions for HW2."""

import matplotlib.pyplot as plt
import numpy as np
from scipy.special import jv
import peakutils
import matplotlib.colors as colors
from matplotlib.colors import SymLogNorm

def problem1():
    D = 10
    l = 0.01

    # Evaluate the power function on a list of phases.
    thetas = np.arange(-0.005, 0.005, 0.00001)
    ps = (np.sinc(thetas * D/l))**2

    # Use PeakUtils to find some peaks.
    # Slice to get the five central peaks.
    idx = peakutils.indexes(ps, thres=0.0001, min_dist=0.001)
    idx = idx[len(idx)/2 - 2:len(idx)/2 + 3]

    # Print out the results, then plot them.
    for i in idx:
        print 'Theta: ', thetas[i]
        print 'Power: ', ps[i]/max(ps)
        print

    plt.plot(thetas, ps, '-b')
    plt.scatter(thetas[idx], ps[idx])
    plt.savefig('problem1.png', dpi=200)


def problem2():
    D = 1
    dD = 0.1
    l = 1
    def Gain(theta, l):
        D = 1
        dD = 0.1
        G = 2 * (np.pi/l)**2 * jv(0, (np.pi * D/l) * theta) * D * dD
        return G


    thetas = np.arange(-np.pi/2, np.pi/2, 0.01)
```

```
46      gs = Gain(thetas, 1)
47      plt.plot(thetas, gs)
48      plt.yticks([])
49      plt.xlabel('Theta (radians)')
50      plt.ylabel('Gain')
51      plt.savefig('problem2a.png', dpi=200)
52      plt.gca()
53
54
55
56  def problem3(vmax=5000):
57      # Begin by setting up the aperture
58      crossbar_width = 3
59      r_circle = 10
60      r_big_circle = 50
61      p0 = 10
62
63      def a(x, y):
64          if np.sqrt(x**2 + y**2) <= r_circle:
65              p = 0
66          elif abs(x) <= crossbar_width or abs(y) <= crossbar_width:
67              p = 0
68          elif np.sqrt(x**2 + y**2) > r_big_circle:
69              p = 0
70          else:
71              p = p0
72          return p
73
74
75      xs = np.arange(-1. * r_big_circle,
76                     1. * r_big_circle,
77                     0.05)
78      ys = np.arange(-1. * r_big_circle,
79                     1. * r_big_circle,
80                     0.05)
81
82      apertures = np.zeros((len(xs), len(ys)))
83
84      for x in range(len(ys)):
85          for y in range(len(ys)):
86              apertures[x, y] = a(xs[x], ys[y])
87
88      ft_real = np.real(np.fft.fft2(apertures))
89      ft_abs = np.abs(np.fft.fft2(apertures))
90
```

```
91
92      fig, (ax1, ax2) = plt.subplots(1, 2)
93      ax1.imshow(apertures)
94      ax1.set_xticks([])
95      ax1.set_yticks([])
96      cax2 = ax2.imshow(ft_abs,
97                        vmin=np.min(ft_abs),
98                        vmax=vmax)
99
100     fig.colorbar(cax2, fraction=0.046, pad=0.04)
101     ax2.set_xticks([])
102     ax2.set_yticks([])
103
104     plt.savefig('problem3.png')
105     #plt.show(block=False)
106     plt.gca()
107
108     # For log color:
109     #ax2.pcolor(ft, norm=SymLogNorm(linthresh=0.1, linscale=0.1, vmin=np.min
110
111
112
113
114
115
116 # The End
```


Planck_func.png